Program Versions

Page application

Fri, Jan 26, 2001

A new page application called VERS supports two utility listings. One compares the version dates of downloaded program files between two nodes. The second merely lists the contents of the entries in the local application table (LATBL). This note provides a few details about this application page.

Page layout

Upon entry, the page looks like this:

```
V PROG VERSIONS   01/26/01 1237
TARG<0509> REF<0562> LIST<0576>
     LA'S
```

The entry fields are the target node#, the reference node#, and the listing node#. To activate the comparison of program files between the target node and the reference node, interrupt anywhere on this line. To activate the listing of the LATBL in the target node, interrupt on the next line with the cursor in the area of the "LA'S" prompt. No other keyboard interrupts produce actions at this time.

Compare program versions function

Upon activation, the program reads the entire CODES table from both the target and reference nodes using a one-shot request for each node in turn. Sufficient patience is included for access to far-away nodes. (If the reply to the request is not immediately forthcoming, it waits up to a second before giving up.) Once the reference node has been read, repeated activations with different target nodes do not again interrogate the reference node until a one minute time-out has occurred, or unless the keyboard interrupt occurs with the cursor in the reference node# field area of the line. Repeated activations, however, always collect the CODES table from the target node.

After collecting the CODES table, which contains a directory of the installed files, comparisons are made of the target directory against the reference directory. For each installed file in the target node, a search is made of the reference directory for the same file. If the file exists in both directories, the version dates are compared. If they are different, a report is generated for that file. If the version dates match, no report is generated, unless the sizes do not match. The file name is appended with a symbol to further illustrate the mismatch. The file is also reported in the case that the file is not found in the reference node.

A count is made of the number of target node files, the number of reference node files, and the number of files reported on the screen and optional listing output.

Here is an example of the listing produced on the screen:

```
V PROG VERSIONS   01/26/01 1229
TARG<0576> REF<0508> LIST<0576>
  42 LA'S  118         22
PAGEPARM>  LOOPTFTP>
PAGEEDAD>  PAGESWFT>
PAGEEDBD>  DATANLIN-
```

```
PAGEMDMP>  DATANDZR-
PAGEVERS-  LOOPBPMP-
PAGELAPP>  DATANPHI-
PAGEECHO>  DATANMSC-
PAGEACRQ>  DATANMRF-
LOOPACLK-  DATANTEV-
LOOPAAUX>  DATANWBL-
LOOPAERS-
LOOPDBDL<
```

The meanings of the attached symbols are as follows:

>	Target version newer than reference version
<	Target version older than reference version
–	Target file does not appear in reference node
*	Version dates match, but sizes differ.

The above example shows that eleven files in node 0576 do not exist in node 0508. Ten files in node 0576 are newer than the corresponding files in node0508. And one file in node 0576, LOOPDBDL, is older than the version in node 0508. The other 20 (unlisted) files in node 0576 are the same versions as those in node 0508.

In addition to the display on the screen, which allows for listing up to 36 reportable files, the listing node option provides a listing from the serial port that has no such limitation. Here is the listing output produced when the above example was used:

```
  PROG VERSIONS   01/26/01 1229
TARG<0576> REF<0508>
PAGEPARM>
PAGEEDAD>
PAGEEDBD>
PAGEMDMP>
PAGEVERS-
PAGELAPP>
PAGEECHO>
PAGEACRQ>
LOOPACLK-
LOOPAAUX>
LOOPAERS-
LOOPDBDL<
LOOPTFTP>
PAGESWFT>
DATANLIN-
DATANDZR-
LOOPBPMP-
DATANPHI-
DATANMSC-
DATANMRF-
DATANTEV-
DATANWBL-
```

Here is another example, for a case when both nodes have the same version:

```
V PROG VERSIONS   01/26/01 1229
TARG<0517> REF<0619> LIST<0576>
  27 LA'S   52          0
```

In this case, no listing node output is produced. There were 27 files found in the target node, and all of them match the versions of the same program file in the reference node, which itself houses 52 files. No listing lines were output.

List LATBL contents function

The local applications table listing function was included in this page application, because much of the functionality was already there in the comparison of versions function.

The entire LATBL from the target node is read. A short-format listing is shown on the screen, for up to 36 entries. A longer-format listing is written to the serial port of the listing node, for any number of entries. The short-format consists of the first part of the long-format listing. Here is an example of the screen following an LATBL listing activation:

```
V PROG VERSIONS   01/26/01 1229
TARG<0517> REF<0619> LIST<0576>
  13 LA'S              13
  1* AAUX   16* DBDL
  2  AERS
  3  FTPM
  4  GATE
  5* SLOG
  6* TFTP
  9* DNSQ
 11* GRAD
 12* CROB
 13* DRIV
 14* PINH
 15* PHAS
```

The total number of the existing LATBL entries is shown, matched by the count of the number of listing output lines. The short-format entries are listed on the screen in three columns. In each case, the entry number, based at 0, is indicated, followed by an asterisk if the enable bit for that entry is set, followed by the 4-character file name. (The prefix LOOP is assumed.) Entry numbers are not listed if they are unused; *i.e.*, no file name has been entered.

The corresponding long-format listing output is as follows:

```
TARG<0517> LATBL  01/26/01 1229
  #  LOOP    smPtr  eBit params
  1* AAUX 0097EEE8  00AE 09EA 0000 0000 0004 0000 0000 0000 0000 0000
  2  AERS 00000000  00A5 09EA 0040 0400 0400 0004 0003 0000 0000 0000
  3  FTPM 00000000  00AC 0000 0000 0000 0000 0000 0000 0000 0000 0000
  4  GATE 00000000  00AB 0000 0000 0000 0000 0000 0000 0000 0000 0000
  5* SLOG 00946E88  00C4 4C3A 83E1 797A 000A 0258 0000 0000 0000 0000
  6* TFTP 00942A90  00C2 0000 0000 0000 0000 0000 0000 0000 0000 0000
  9* DNSQ 0097B920  00C0 0003 0002 7000 0000 0000 0000 0000 0000 0000
 11* GRAD 0097B860  03AF 0302 039F 0000 0000 0000 0000 0000 0000 0000
 12* CROB 0097A0B0  03AE 0302 0305 0315 030F 0317 030B 0316 0395 0397
```

```
13* DRIV 0097A018  03AD 0310 031E 031F 031A 0316 0397 0395 0342 0391
14* PINH 0097F230  03AC 0302 030F 0315 0000 0000 0000 0000 0000 0000
15* PHAS 0097B2F8  03AB 0302 0300 0310 009F 0000 0000 0000 0000 0000
16* DBDL 0093CC40  00D2 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

The first part of each line is the same as in the short-format listing on the screen. Next is the static memory pointer currently in use by the active local application for this entry. Each local application instance allocates some static memory for use in maintaining its context across successive calls. This is needed because a local application is merely a function that is called at 15 Hz and upon other more specialized times. The static memory pointer is passed as an argument to the local application each time the call is made. Finally, the listing includes the 10 words of parameters, which are also passed to the local application each time it is called. The other 9 parameter words may be constants, analog channel numbers, or binary bit numbers, depending on the needs of the particular local application.

The first of the 10 parameter words always has the significance of an enable bit number for the local application instance. When the enable bit transitions from a zero to a one, the instance is activated. Upon activation, the local application allocates its static context memory and reserves any other system resources it needs. During the time that the instance remains activated, the system calls the local application at 15 Hz, during Data Access Table processing. When the enable bit transitions to a zero, the instance is deactivated, and the local application releases all of its resources. (Multiple instances can be used for the same local application, in which case the parameter values and a separate context memory distinguish each instance.)

The most common special call is used for an application that uses the network, in which messages destined for it are passed to it via a call. A simple example of this is the LOOPECHO application, which supports the test UDP port 7 ECHO protocol. Thus LOOPECHO amounts to a UDP ECHO protocol server.

The LATBL listing gives an overall view of the LATBL. Page E, using the program PAGELAPP, is the usual means of adding/modifying entries in the LATBL, but it only allows viewing one entry at a time. This was the motivation for adding this overview listing option to PAGEVERS.